Information Extraction using Rule based Software Agents in Knowledge Grid

Praveen Desai[1],Dinesh Acharya[1], and Ashalatha Nayak[1]
[1]Manipal Institute of Technology,Manipal, India
Email: praveen.desai@manipal.edu

***Abstract*—For the successful information processing and handling of document collections, effective information extraction methods are necessary. A distributed team work environment requires team knowledge management. A knowledge flow exists in team work processes and this knowledge flow reflects the knowledge level cooperation in team work, which in turn defines the effectiveness of team work. Distributed software development team focuses on work co-operation and resource sharing between members during software development life cycle and knowledge flow should reflect cognitive cooperation process dynamically. Hence each team member can use experience of predecessor accumulated during previous projects and avoid redundant work. With the advent of the networks, the system specification is done in one geographic area and the design in some other place. The entire software development process has distributed resources such as five generic up-level ontologies and a knowledge based [KB] issues and solutions ontology. An issue and solution pair criteria is based on organizational goals, priorities, cost and timeliness. In this paper, we present the challenges in distributed team environment and information extraction mechanisms focusing on text marker system and its applications.**

Index terms: Agents, Knowledge Grid, Message passing

I Introduction

In today's knowledge era, information networks play an important role in communicating in distributed environment [1].These information networks support the critical infrastructure that is responsible for much of the productivity behind our economic growth. The effectiveness of a team work not only depends on individual's knowledge; depends on cooperation and precise communication among them. The drawbacks of remote communication as compared with one-to-one communication are a major challenge for geographically simulation, problem solving, and decision making by using sharable knowledge. The basic extraction modules are viewed as IE predicates and users define whether these IE predicates satisfy certain properties. Also, text-specific operations and present optimization techniques are included using an algebra that that exploit properties of new operators[2].

The purpose of the Knowledge Grid is for sharing and managing globally distributed knowledge resources in an efficient and effective way.

It incorporates epistemology and However, the issues like what job to be completed, what problems have been raised and clarified, clarity in project scheduling , availability of resource and to make on time decision are the major concerns of a software life cycle. Consequently, these issues cause project delay as well as anxiety among team members. The communication can be achieved easily by aggregating the collective knowledge about the project, the domain knowledge and skillsets of managing the task into a common platform with the help of intelligent agents and allow them to share the repository called Knowledge Grid.

II. Related Work

Information Extraction is a mature area of research that has received widespread attention in the NLP, AI, web and database communities. The methods like rule-based approaches [3] and machine learning based approaches have been proposed. To improve the quality of results were focussed in related work available.

A. Frameworks for IE

The NLP community has developed several software architectures, such as GATE, ATLAS, and UIMA . The focus of these architectures is to provide a framework where annotators developed by different providers can be integrated and executed in a workflow based structure.[4][5]

B. Optimizing IE

Recently, there has been work on improving the efficiency of information extraction tasks in specific settings. The techniques for scaling up the named-entity recognition and a technique to prune documents considering the challenges of efficiently composing multiple extraction modules into a larger program that enables people or agents to effectively generate, capture, publish, share the information.[6][7]

III. Agent Decision Making

Interaction is the main driver for agent decision making. The difference between *reasoning* and *decision making* is that reasoning is based on "single thread of control" practice [8]. That is, in reasoning only one decision making entity is present and active. However, in decision making, multiple decision making entities are running simultaneously and independently where outcome of one action may affect the other decision making process.

An agent's decision making process is summarized as follows[9]:

1) collecting relevant data from the other agents.
2) reframing and interpreting data/information.
3) Identifying the interaction class and appropriate decision making methods.

DOI: 03.LSCS.2013.1.526

4) Building the representation model based on the interaction class, using different classes of games in game theory, utility theory or other uncertainty management theories.
Figure1 depicts the scenario where agents decision making helps in distributed environment.

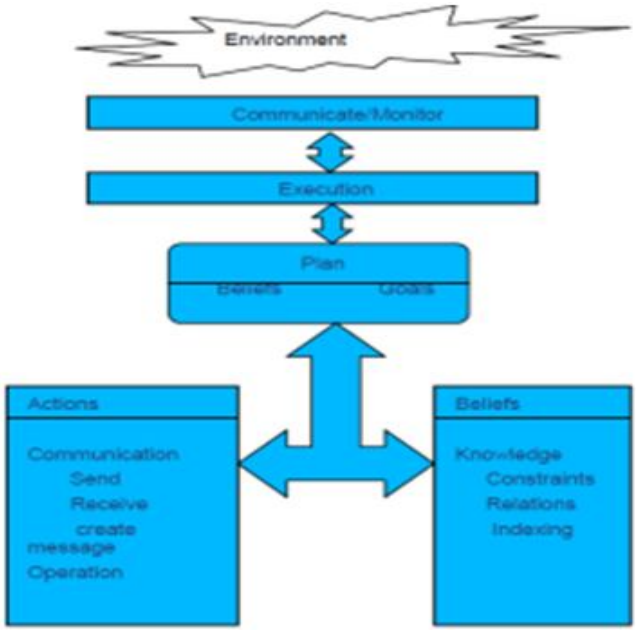

Figure 1 Service Agent \model concept

Software agent consults knowledge specified in ontology as well as in knowledge base [10]. The ontology is a computer readable description of knowledge that describes objects such as components, documents, projects et al and their respective instances stored in databases. Such enumerated knowledge used by agents for getting answers from user queries, making decisions, conveying results autonomously.

The Environment is the outside world in a distributed scale.

- This could be invokable applications which are interacting with the system
- A human interface that is part of work place environment
- autonomous activity that implements the tasks without human intervention

The plan of service agent encapsulates the business logics of how to use the Web service operations to achieve a certain business goal, how Web service operations can be integrated. Each plan denotes one of capabilities that the service agent has.

The Execution and Communication component helps the agent to communicate and react with the environment.

- include issues such as node reputation, ranking and management of large-scale open systems.
- provide implementation methods and common platform, enabling the easy creation of infrastructures
- Common protocol for discovery and communication between heterogeneous services.

The service agent model is composed of four fundamental elements that are *Beliefs*, *Actions* and *Plans*.

Beliefs represent the current state of the agent's internal and external worlds.

Actions are the set of actions that the service agent is able to perform.

Plans are the set of plans that the service agent has. Each plan is a set of activities that is executed as per goals defined. Each plan denotes one of the capabilities that the service agent has. The plan is defined as tuple (*Os, Ra, Goal*), where:
a) *O*sis a set of Web service operations.
b) *Ra* is a set of relations in Web operations *Os*, *Ra* = DiCj, where *Di* and *Cj*are data flows set and control flows set, respectively.
c) *Goal* is the business goal that the plan achieves, which is denoted as tuple (*entry*, exit), where *entry*and *exit*denote the input and output parameters of the Plan respectively.

During the process of service composition, the plan of a certain service agent can be one of three statuses that are *Unexplored*, *Exploring* and *Explored*. The status *Unexplored* means that the plan has not been searched, the status of *Exploring* means that the plan is being searched and the status of *Explored* means that the search for the plan has been finished.

A. Dependence Relation [DR]:

An agent is said to be dependent on another if the latter can help to achieve its goal. Considering Agent Services Si and Sj such thatSi={Ai,Bi,Pi} ;Sj={Aj,Bj,Pj}

If PLAN 'i' of Si is dependent on PLAN 'j' of Sj, then dependency relation DR (Si,Pi,Sj,Pj,x)
where x-> parsing of [Goal Pi) ∩Goal Pj].

If Goal Pi = Goal Pj => independent Relationelse dependence Relation.

Using DR, the dependence graph [DRG] /directed graph among service agent can be constructed as DRG{ V, E}where V is a vertex or node and E is Edge or a link.

The proposed methodology includes following 3 steps.
Step1 Initialization

When a service requirement is submitted, it creates a *user agent*. It sends a message *mBroadcast*(Ua, *ri, ro*) to all service agents to notify that a new task arrives. When the service agent *Sai*receives the message *mBroadcast*. The service agent *Sai*checks whether there is a plan *p* whose output parameters can help the user agent to achieve the service requirement. If it is true, it sends a message denoted as *Provide*(*Sai*, *p, ro*) to the user agent with the aim to tell the user agent that service agent *Sai*can provide the parameters set *ro*)" *GetGoalOutputs*(*p*) for the user agent by the output parameters of its *plan*p.
Step 2 Tracing Phase

This step is to construct the dependence graph based on the dependence relations among service agents. The user agent checks whether the set of the minimal cover solutions is empty and notify that the requirement can be completed or not. Then the minimum length is searched for sending request messages.

Once the service agent *Sai*receives *mRequest*(*s, p', Sai, p, x*) message, following algorithm is executed.

DOI: 03.LSCS.2013.1.526

According to the status of the Plan *p*, we can identify 2 situations like the status of *Unexplored*, which means the search for the plan has not been carried out before and it is the first time that the service agent receives the request message about the Plan *p* and second one as the status of *Explored*, which means that the search for the Plan *p* has been finished.

Step 3 Forward Phase: When service agent Sa receives the Response Message RSm, it forwards the same to other Sa. Thus the communication among peer Sa's will be established. The sample semantic mapping is given below

```
//Algorithm: KSemantic Matching
Begin
Switch(e.type){
        Case start:
        Create a plan for each action class
        Case class:
        Plan :=profileBelief.getAction(e.name )
        If (plan is null)
        Goal:=false
        Else{
CName.table+=(e.plan)
        Goal:=true
        }
        BeliefList :=getClassBelief(e.beliefname)
        If ( Belieflist Not null) {
        For each class Ci in BeliefListdo{
        Plan :=profileBelief.getAction(Ci )
         If match is not null {
        CName.table+=(e.plan)
        Goal :=true
         }
        }
        }
Case end:
GenerateResult ()
}
End
```

IV. An Algebraic Approach

In this section, we describe a simple object-relational data model for representing annotations over a given document along with a set of logical operators and demonstrate rule-based annotators as compositions of these operators.[11][12]

A. Data and Execution Model

The algebra is designed here to extract annotations from a single document at a time, and we define the algebra's semantics in terms of the current document being analyzed. The current document is considered as a string called *dText*, identified by each annotator that satisfy a set of rules and marks each region with an object called a *span*. A span is simply an ordered pair h*begin*, *end*ithat denotes the region of *dText* from position *begin* to position *end*. For example, if *dText* was the string "Informationretrieval", *h13, 22i*: "retrieval" would denote the range from characters from positions 13 to 22 of the document. A simple relational data model is based on algebra with three attributes: span, tuple, and relation. In data model, a *tuple* is an finite sequence of n spans hs1, ...,swi; where n is *width* of the tuple. The span operators available in Operator Algebra is shown in Table I [13]. A *relation* is a multiset of tuples of same width as a constraint. Each operator here takes zero or more relations as input and produces a single output relation.

Table I. Meaning of span operators using Operator Algebra

SPAN PREDICATES	Explanation
Predicate s1 _d s2	s2 follows s1,and both are mutually exclusive and maximum *d* characters between the end of s1 and the beginning of s2
s1 ≃s2	the overlap between s1,s2
s1 ⊂s2	s1 is strictly contained within s2
s1 = s2	spans are identical

To annotate all the documents in a global annotation database, the framework applies an algebra expression to every local annotation repository or database DB separately and execution proceeds as follows:

```
        E⟵ {algebra expression }
for localDBin globalDBdo
begin
1. { ReadlocalDB into main memory }
2. R⟵ E(localDB)
3. { Add R to localDB}
4. { Write modified localDB to disk }
end
```

To run multiple annotators, step 2 in the above process can be repeated multiple times per document.

At the beginning of an OWL document, a start event is generated for a particular class C and individual I of that class. When requested property is found, the triplet [P, A, B] property is generated where P is the name of the PLAN, A is specific activity to perform as ACTION and B is a range or constraint defined for the activity, BELIEF. At the end of an OWL document/result, an end( ) event is generated[13]. Considering the following queries,

```
Q1 = (?x type Technique)
Q2=(?x type Schema)(?x Method ?y)
Q3=(?x type Solution) (?x 'C++')
```

The corresponding AgentProfileHash table and Agent Match table is shown in Table II and Table III respectively.

Table III. Agent Profile Hash

&1	Technique
&2	Schema
&3	Solution
&4	Method

Each entry of the second level of *Agent Profile Hash* contains the attributes such as *ID* which generates unique identifier, *NextMatch* denotes the semantic connection of already available triplet, *Value* indicates the probable values

DOI: 03.LSCS.2013.1.526

TABLE III: AGENT MATCH TABLE

ID	NextMatch	Value	Key
1	{&3}		P
2		'C++',JAVA	x, y
3	-	OOP	A

as per the query and *Key* specifies the variables used. Thus all four attributes define the *AgentProfileHash* event.

The procedure for building AgentProfileHash is given below:

Procedure AgentProfileHash G

Begin

```
for each triple Ti in Goal G do{
tripleInfo:=AgentProfileHash.getTriple(Ti.Goal)
If (tripleInfo is Null)
tripleInfo:=profileHash.insertTriple(Ti.Plan)
tripleInfo:=profileHash.insertTriple(Ti.Action)
tripleInfo:=profileHash.insertTriple(Ti.Belief)
tripleInfo.insertTripleInfo(G.id, Ti.info)
}
End;
```

For each Goal Gi in *AgentProfileHash*, if the entry for the NextMatch is null, the matchTable becomes the result for Goal Gi, else the result for Goal Gi is obtained by joining all the match tables of NextMatch attributes while parsing these attributes.

Procedure AgentHashResult

Begin

```
for each Goal Gi in AgentProfileHash do{
for each Plan Pi in the entry of Gi do {
if (Pi.NextMatch is NULL)
        Gi.result :=Gi.Goal
        else {
        Gi.result :=Gi.Pi
while (Gi.NextMatchisnot null){
Gi.result := Gi.result U Gi.NextMatch.Action
Gi.NextMatch++
}
}
}
```

end

Thus the algebraic approach will facilitate the information extraction from the repository using agent mechanism for ease of operation.

CONCLUSION

The ability to extract desired pieces of information from natural language texts is an important task with a growing number of potential applications. Tasks requiring locating specific data in team work or web pages are particularly promising applications. Manually constructing such information extraction systems is a laborious task; however, algebraic methods have the potential to define the span of the development process. The patterns employ limited syntactic and semantic information to identify potential slot fillers and their surrounding context.

ACKNOWLEDGMENT

The authors would like to thank parent institution for their support and necessary guidance along with anonymous reviewers for their constructive comments and suggestions during the conference.

REFERENCES

[1] Turmo, J., Ageno, A., Català, N.: Adaptive Information Extraction. ACM Comput.Surv. 38(2) (2006)

[2] Appelt, D.E.: Introduction to Information Extraction. AI Commun. 12(3) (1999) 161–172

[3] Ferrucci, D., Lally, A.: UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. Nat. Lang. Eng. 10 (2004) 327–348

[4] Kluegl, P., Atzmueller, M., Puppe, F.: A Framework for Semi-Automatic Development of Rule-based Information Extraction Applications. In: Proc. LWA 2009 (KDML – Special Track on Knowledge Discovery and Machine Learning). (2009)

[5] Ciravegna, F.: (LP)2, Rule Induction for Information Extraction Using Linguistic Constraints. Technical Report CS-03-07, University of Sheffield, Sheffield (2003)

[6] Kluegl, P., Atzmueller, M., Puppe, F.: Meta-Level Information Extraction. In: The 32nd Annual Conference on Artificial Intelligence, Berlin, Springer (September 2009)

[7] Cunningham, H., Maynard, D., Tablan, V.: JAPE: A Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, University of Sheffield (November 2000)

[8] Gurevych, I., Müller, M.C.: Information Extraction with the Darmstadt Knowledge Processing Software Repository (Extended Abstract). In: Proceedings of theWorkshop on Linguistic Processing Pipelines, Darmstadt, Germany (Jul 2008)

[9] Ogren, P.V., Wetzler, P.G., Bethard, S.: ClearTK: A UIMA Toolkit for Statistical Natural Language Processing. In: UIMA for NLP workshop at LREC 08. (2008)

[10] F. Ciravegna. (LP) 2, Rule Induction for Information Extraction Using Linguistic Constraints. Technical Report CS-03-07, Department of Computer Science, University of Sheffield, Sheffield, (2003)

[11] Martin Atzmueller, Peter Kluegl, and Frank Puppe. Rule-Based Information Extraction for Structured Data Acquisition using TextMarker. In Proc. of the LWA-2008 (KDML Track), pages 1–7, (2008)

[12] Mary Elaine Califf and Raymond J. Mooney. Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction. Journal of Machine Learning Research, 4:177–210, 2003.

[13] Atzmueller, M., Nalepa, G.J.: A Textual Subgroup Mining Approach for Rapid ARD+ Model Capture. In: Proc. 22nd Intl. Florida AI Research Soc. Conf. (FLAIRS), AAAI Press(2009)

DOI: 03.LSCS.2013.1.526